プログラム動的入替えのための方式検討

1V－7

伊藤　健一
ＮＴＴデータ通信（株）

谷口　秀夫
九州大学　工学部

１．はじめに

近年、２４時間連続のサービスをシステムに要求する声が高まっている。故障をシステムの外に見せない等の信頼性向上の研究や、保守作業開始までの時間短縮のためのリモートメンテナンス等の検討がハードとソフトの両面からなされている[1][2]。我々は、ダイナミックリンク機能（ＤＬと略す）を利用して、サービスを実現している応用プログラム（ＡＰと略す）を動的に入替える方法について研究を進めている。既に、ＡＰを動的に入替える際の条件について報告している[3]。本稿では、プログラムを入替えるための方式検討について報告する。

２．プログラムの入替えの条件

プロセスを存在させたまま、プロセスのテキスト領域やデータ領域の一部を入替える「プロセス継続型」の入替え方法にＤＬを利用するためには、２つの条件がある[3]。１つは、プログラム構成に関する条件である。もう１つは、入替えの対象となるロードセグメント（ＬＳと略す）とプロセスの位置関係による条件（表１）である。位置関係は、以下のように定義する。

未使用：入替え対象のＬＳを実行する前か、既に実行を終えリターンした状態。
走行中：入替え対象のＬＳ自体を実行している状態。
呼出中：入替え対象のＬＳから更に他のＬＳを呼出し、実行している状態。呼出し先の処理が終了後、対象のＬＳに実行が戻る。

表１　位置関係による入替え条件

関係	条件
未使用	(a) 入替えたＬＳのアドレス解決が必要 (b) 入替えたＬＳの外部変数の値の保証（引き継ぎ）が必要
走行中	入替えは不可能
呼出中	(a) 入替えたＬＳのアドレス解決が必要 (b) 入替えたＬＳの外部変数の値の保証（引き継ぎ）が必要 (c) 別のＬＳを呼び出したアドレス（戻りアドレス）を変更しない (d) 別のＬＳを呼んだ関数の内部変数を変更しない

３．入替え可能状態の把握

プロセスとＬＳの位置関係が前節で述べた３つのいずれの状態であるかを判定する方法が必要である。各プロセスについて、そのプログラムカウンタ（ＰＣと略す）の値を見ることにより、それぞれのプロセスがどのＬＳを走行しているかは判断できる。ＰＣが入替え対象のＬＳの中にあれば「走行中」である。しかし、ＰＣを使って状態の遷移を把握することは計算機の負荷が大きくなる。例えば、命令トレースによりＰＣの値を常にチェックすることになってしまい、プロセスの数が増えるにつれオーバーヘッドが膨大になる。

そこで、ＬＳの利用状態を把握するために、以下の２つのフラグを導入する。

(1) sflag：当該ＬＳが他のＬＳを呼出している数をカウントする
(2) rflag：当該ＬＳが他のＬＳから呼出されている数をカウントする

sflagは初期値０とし、プロセスが他のＬＳを呼出すと加算し、リターンすると減算する。つまり、フラグの値は当該ＬＳからの呼出しの数であり、１以上なら「呼出中」のプロセスが存在する。rflagも初期値０とし、当該のＬＳが他のＬＳから呼出されると加算し、リターンしたら減算する。状態遷移の契機は以下のようになる（図１）。

(1) rflagが加算されて、０でなくなる時
　未使用　→　走行中
(2) rflagが減算されて、０になる時
　走行中　→　未使用
(3) sflagが加算されて、０でなくなる時
　走行中　→　呼出中
(4) sflagが減算されて、０になる時
　呼出中　→　走行中

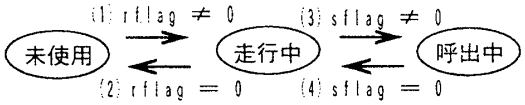

図１　状態遷移とフラグ

A Method of Dynamic Program Restructuring.
Ken'ichi ITO* and Hideo TANIGUCHI**
*: NTT DATA Communications Systems Corporation
**: Kyushu University

表２　フラグによる状態把握

	フラグ状態	走行中	呼出中	入替え条件
状態１	rflag = sflag = 0	ない	ない	(a), (b)
状態２	rflag > 0, sflag = 0	ある	ない	(a), (b), (c), (d)
状態３	rflag = sflag > 0	ない	ある	(a), (b), (c), (d)
状態４	rflag > sflag > 0	ある	ある	(a), (b), (c), (d)

rflagとsflagにより、当該ＬＳを走行中のプロセスの数（rflagからsflagを減じた値）と、他のＬＳを呼出しているプロセスの数（sflagの値）が把握できる。この２つのフラグにより、それぞれのＬＳがプロセスとどのような位置関係にあるかが、ＰＣを利用せずに簡単に判断できる（表２）。

４．実現方式

４．１　プロセスの制御

ＬＳを入替える処理は、以下の３つに大きく分類できる。入替え可能状態の保持処理は、入替え処理の間に当該ＬＳを利用するプロセスが１つも走行しないように制御すればよい。しかし、プロセスの走行を停止させることは、サービスに影響がでるため、当該ＬＳ以外を利用したプロセスの走行を許す必要がある。

(1) 入替え可能状態の保持処理
　走行中の状態へ遷移するプロセスの動きを遷移の直前で停止させる。
(2) 入替え可能状態への移行処理
　走行中の状態へ遷移するプロセスの動きを遷移の直前で停止させるとともに、走行中の状態のプロセスがなくなる契機を把握する。
(3) 入替え処理
　ＯＳが持つプロセス管理やメモリ管理の機能を利用して、当該ＬＳを入替える。

４．２　ＬＳの状態管理

ＬＳの状態を管理するには、次の３つの処理をＡＰとＯＳで分担することが必要である。

(1) 呼出しやリターンの契機を検出する
(2) rflagやsflagの値を加算または減算する
(3) 状態が遷移したことを検出する

処理(1)は、関数呼出しをＯＳで検出するよりも、ＡＰの関数呼出部分に手を加える方が実現が容易である。例えば関数名をマクロ化して、関数呼出しの前後に検出処理を行なうようにする。処理(2)は、ＡＰ、ＯＳのどちらで行なうことも可能であるが、ＡＰプログラムへの追加を少なくするためＯＳで行った方がよい。処理(3)は、入替え可能な状態への移行や保持を行うためにプロセスを制御するため、ＯＳで実現するべきである。従って、処理(1)はＡＰ、処理(2)と処理(3)はＯＳの分担が好ましい。

４．３　ＬＳの入替え処理

ＯＳはＬＳの状態管理とともに、ＬＳの入替えを要求システムコールにより、ＬＳの入替え処理を行なう。入れ替えの処理方式は、入替え処理を行なう契機と、その処理を実行するプロセスコンテキストで以下のように分類できる。

(方式1) 入替え要求システムコールを発行したプロセスのコンテキスト内で、なるべく早く入替える。
(方式2) 当該ＬＳを利用するプロセスのコンテキストになったら、なるべく早く入替える。
(方式3) 入替え要求が発生してもすぐに入替えを行なわず、当該ＬＳが実行される直前に、当該ＬＳを利用するプロセスのコンテキストで行う。

ＬＳの入替え要求が発生した時の、当該ＬＳの状態により、有効な方法が異なる（表３）。ＬＳが「未使用」か「呼出中」の状態にあり、直ちに入替えが可能な場合には、(方式3)が有効である。一方、ＬＳが「走行中」の状態で、直ちには入替え不可能だが、その後可能になる場合には、(方式2)が有効である。

表３　入替え方式の比較

	即実行可能		即不可能、後に実行可能	
	複雑さ	遅れ	複雑さ	遅れ
方式１	×	○	×	○
方式２	△	○	△	◎
方式３	◎	△	○	△

５．おわりに

本稿では、ＬＳをプロセスを継続したまま入替えるときに、対象となるＬＳとプロセスの関係を把握する方法について提案し、それを実現するための処理について検討した。今後、基本的な入替え機能を実現し、提案方式の有効性を確認する予定である。

参考文献
[1] Parag K. Lala（当麻監訳）：「フォールト・トレランス入門」、オーム社（1988）
[2] 萩原他：特集「フォールトトレラント分散システム向けアルゴリズム」、情報処理、Vol. 34, No. 11, pp. 1335-1374 (1993)
[3] 伊藤、谷口：「ＡＰの動的入替えに関する一考察」、第48回情処全国大会（1994）